B_025

モデル検査器を用いたFUCEマルチスレッドプログラムの開発
Developing FUCE Multithreaded Program with a Model Checker

越村 三幸* 梅田 眞由美† 平兮 亮* 藤田 博*
Miyuki Koshimura Mayumi Umeda Ryo Hirana Hiroshi Fujita
長谷川 隆三*
Ryuzo Hasegawa

1 はじめに

FUCE（*Fu*sion of *C*ommunication and *E*xecution）[1]はマルチスレッド処理アーキテクチャで，そのプロセッサはMIPS命令を拡張したアセンブリ言語を実行する．これまでのFUCEプログラミングは，このアセンブリ言語で行う必要があった．本稿では，モデル検査器SPIN（*S*imple *P*ROMELA *I*nterpreter）[2]を用いて並行プログラムを開発し，それをデバック・検証した後，FUCEアセンブリ言語に変換する手法を述べ，ソーティングプログラムにこの手法を適用した事例を報告する．

2 FUCEプログラム

FUCEプロセスは，一つ以上のスレッドから構成される．スレッド間に依存関係がなければスレッドは並列に実行されうる．図1にスレッド間の依存関係の例を示す．(a)は三つのスレッドA，B，Cの依存関係を示しており，BはAの結果を，CはBの結果を必要としている状態を表している．FUCEでは，これら三つのスレッドを実行するために，AはBに，BはCに計算結果と共に通知を送る必要がある．この結果の通知を継続と呼び，AはBに，BはCに継続する，と言う．(b)は継続するスレッドが複数の場合を示している．スレッドBとCには依存関係がないので，並列実行が可能であり，DはBとCから継続されないと実行できない．

あるスレッドに対して継続される元のスレッド数をfan-in値と呼ぶ．(b)において，スレッドBのfan-in値は1，Dのfan-in値は2である．

FUCEのスレッド実行制御はすべて継続によって決まり，スレッドは継続されるたびに自fan-in値を1減じ，値が0になると，実行可能状態になる．一旦，実行を開始すると，そのスレッドは実行終了命令を実行するまで，中断することなく走り切る（走り切りスレッド）．

このようにFUCEプログラムは，（走り切る）スレッドプログラムの集合として記述される．

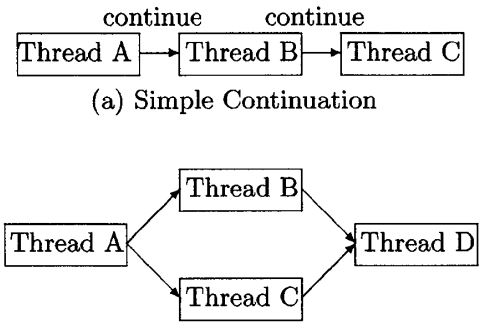

図1: スレッド間の継続

3 SPIN

SPINは，PROMELA（*Pro*cess *Me*ta-*La*nguage）言語で記述したモデルを検証するシステムである．モデルは，プロセスの群として定義される．PROMELAはC言語に似た構文を持ち，CSP（*C*ommunicating *S*equential *P*rocesses）のようなプロセス間の通信による同期機構を備える．検証項目はLTL（*L*inear *T*emporal *L*ogic）論理式で表現する．

SPINには，シミュレーションモードと検証モードがある．シミュレーションモードでは，システム内部の乱数のタネに従うスケジューラによってプロセスの実行順が決定される．このタネを変更することによって，様々なタイミングのシミュレーションが可能である．

一方，検証モードではプロセスのあらゆるスケジュールの可能性が網羅的に検査される．したがって，検証に

*九州大学，Kyushu University
†九州大学（現在，パナソニック ITS(株)），
Kyushu University(Presently at: PASITS)

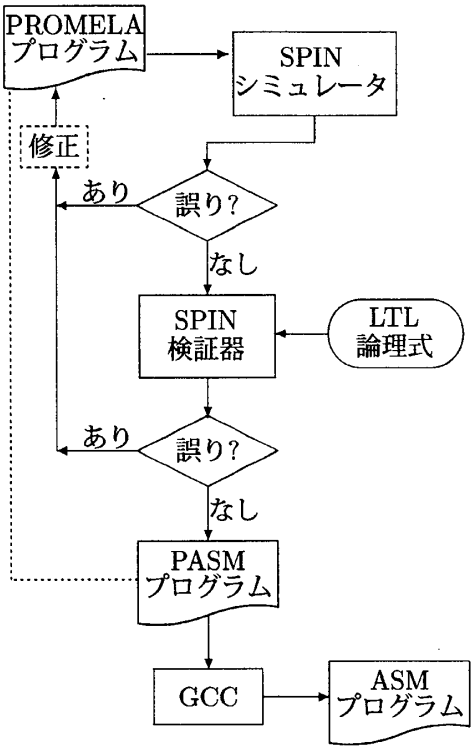

図 2: プログラム開発の流れ

よって誤りが検出されなければ，プロセススケジュールのタイミングによるバグはないことが保証される．

4 FUCE プログラムの開発

図 2 に SPIN を用いた FUCE プログラムの開発工程を示す．図中，ASM は FUCE アセンブリ言語，PASM は擬似アセンブリ言語を表す．ASM はアセンブラーにより機械語に翻訳され，FUCE シミュレータによって実行される．PASM の構文は，C 言語に準拠しており MIPS 用 GCC によりコンパイルが可能である．FUCE 固有の命令は，インラインアセンブラによって PASM プログラムに埋め込まれる．

SPIN を利用したプログラム開発では，最初に，PROMELA でプログラムを記述する．これは，目標とする PASM プログラム（以下では，目標プログラム）のいわばプロトタイプに相当するもので，以下ではこれをモデルと称する．FUCE のスレッドは，PROMELA のプロセスによってモデル化し，スレッド間の継続による同期は，チャネルによるプロセス間通信によってモデル化する．そして，このモデルのデバックを SPIN のシミュレータを用いて行う．シミュレータでバグが検出されなければ，SPIN 検証器を用いて検証を行う．

検証でバグが見つからなければ，（検証済み）モデルを元に目標プログラムを記述する．一般に，目標プログラムから何かを捨象したものがモデルなので，この記述作業は，その捨象した何かの再構成となる．

試みとして，クイックソートプログラムを例題に，モデルと目標プログラムの記述実験を行った．通常のソートプログラム（通常版）とストリームによるパイプライン処理を利用したソートプログラム（ストリーム版）の二種類を用いた．

検証項目として，「いつかソートされた状態になる」と「ソートされた後，その状態が保たれる」を設定した．LTL 論理式として，前者は $\Diamond sorted$，後者は $\Box(sorted \rightarrow \Box sorted)$ と表現できる．ここで $\Diamond$ と $\Box$ は共に様相演算子で，$\Diamond$ は「いつかは」を表し，$\Box$ は「常に」を表す．また，$sorted$ はソート状態を表す．

ここで，「どんな数列を入力しても必ずソートできる」ということが証明されるわけではではない，ことに注意を要する．SPIN ができるのは，具体的に与えた数列に対して，LTL 論理式が成り立つかどうかの判定だけである．今回の検証では，通常版は長さ 13 まで，ストリーム版は長さ 6 までの具体的な数列を与えて検証を行った．数の並び方にもよるが，これより長い配列では，検証時間が 10 分では収まらない場合もあった．

5 おわりに

この研究の開始時点では PASM 言語がなく，FUCE プログラミングは ASM 言語で行われており，プログラミングが気軽にできるという状況ではなかった．何らかの高級言語でプログラムを記述し，そのレベルでデバッグが済んだプログラムを元に ASM プログラムを記述すれば，ASM レベルでのデバッグの負荷が軽減できるのではと考え，高級言語として PROMELA を選択した．

検証機能は措いたとしても，PROMELA は並行システムのプロトタイプの開発には優れた言語であることが，本研究で実感できた．現在，PASM よりも上位の言語の設計を行っているが，その上位言語のデバッグにも SPIN を利用したいと考えている．

参考文献

[1] S. Amamiya, M. Izumi, T. Matsuzaki, and M. Amamiya, "The Fuce Processor: The Execution Model and The Programming Methodologies," PDPTA'06, 2006.

[2] G. J. Holzmann, "THE SPIN MODEL CHECKER," Addison-Wesley, 2003.